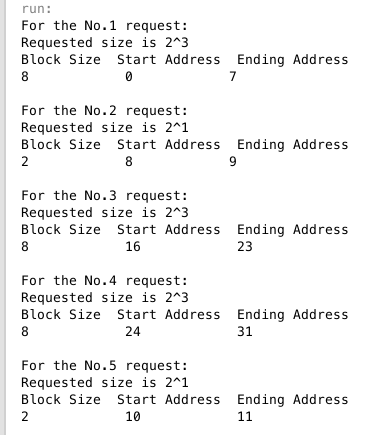
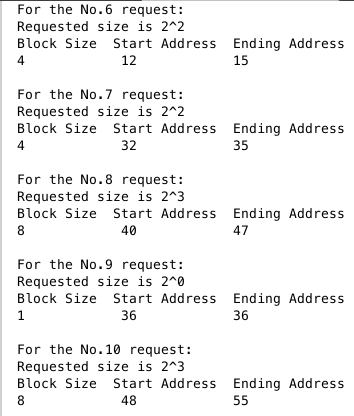
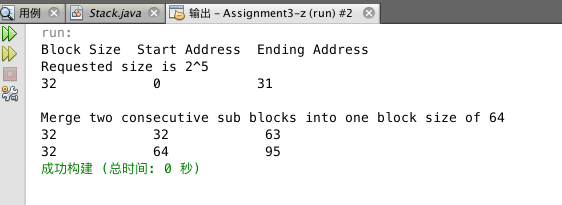
**INFO6205 Assignment 3 Report**

In our program, First, we instantiate a block of which the size is 128, and the block usage management is simulated with binary tree data structure. To follow the requirements, we bias our random block request size toward the lower of the binary tree, which means block size is no more than 16. We randomly request a size block from the memory pool. After 10 requests, let’s see the printed result.

We can see that the returned blocks’ size exactly match the request size and also there is no address conflicts between returned blocks.

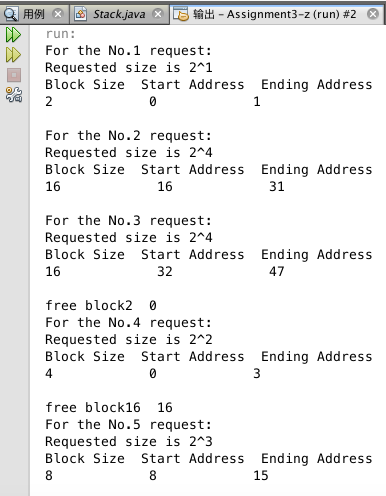
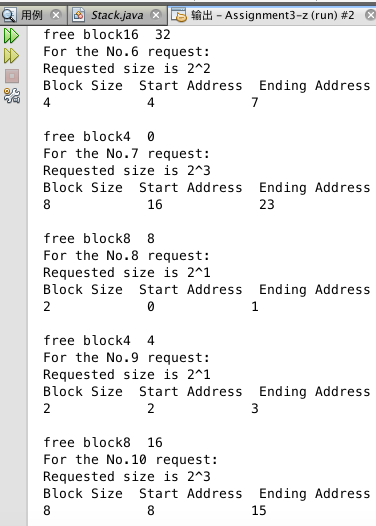
We also implement an approach for reservations of consecutive blocks that on opposite sides of the binary tree (right most blocks combined with the left most block) that happen to be adjacent.



We can see that in our test case, first we request a 32-size block, and then merge two consecutive sub blocks into one block size of 64.

The returned result exactly match the request size and also there is no address conflicts between returned blocks.

After setting a timer for each block of which value is 3, which means that after 3 requests, the previous requested block would be set free again. The result is as follows.

Then let’s see the complexity of our method.

To response to a block request, we first conduct the pre-order traversal through our complete binary tree to find the block of the same request size and also of which the status is “unused”. If the first two requirements are satisfied, we then check the availability of the block which means to make sure all of its descendants’ status are “unused”.

The complexity of pre-order traversal is O(m), and m refers to the number of nodes in the binary tree. Considering we request a block size of n, then the total number of nodes of the block tree is:

1+2++…+

Since n=,

Hence.

Therefore, the complexity of pre-order traversal is O(2n-1) = O(n).

When it comes to the complexity of checking all of a block’s descendants’ status, we can simplify the process to searching the first “used” descendant of the block. Assuming the size of the target block is m, then the number of its descendants is:

2++…+

Since m=,

Hence.

Therefore, the complexity of checking all of a block’s descendants’ status is O(2m) = O(m). The Arithmetic mean of m is:

AM =

*p* is the total number of nodes of the block tree, represents each node’s block size. As discussed above, we request a block size of n, then the total number of nodes of the block tree is 2n-1. And also the sum of each level of the tree equals to *n*. Therefore, we can solve the equation:

AM =

=

= O().

To combine those two methods, we can conclude that the complexity of our request method is O().